

AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

- B1
- 1 1. (Currently amended) A method for verifying type safety of an
2 application snapshot, the application snapshot including a state of an executing
3 program that is moved from a first computing device to a second computing
4 device across a network in order to continue execution on the second computing
5 device, the method comprising:
6 receiving the application snapshot from the first computing device on the
7 second computing device, wherein the application snapshot contains dynamic
8 variables and defines the dynamic state of the executing program and wherein the
9 application snapshot includes a subprogram, an operand stack, and a point of
10 execution;
11 restoring the state of an object within the application snapshot on the
12 second computing device by changing a pointer from an address of the object on
13 the first computing device to an address of the object on the second computing
14 device;
15 examining the application snapshot to identify the subprogram and the
16 point of execution within the subprogram;
17 examining the subprogram to determine an expected structure of the
18 operand stack at the point of execution;
19 validating that the state of the application snapshot on the second
20 computing device is consistent with the expected structure of the operand stack;
21 and
- Sub C1

22 if the state of the application snapshot is validated as consistent with the
23 expected structure of the operand stack, resuming execution of the application
24 snapshot on the second computing device.

1 2. (Original) The method of claim 1, wherein examining the subprogram
2 to determine the expected structure of the operand stack at the point of execution
3 involves examining the subprogram with a code verifier, wherein the code verifier
4 ensures that:

5 the subprogram does not cause the operand stack to overflow and
6 underflow;

7 a use of a local variable does not violate type safety; and

8 an argument of an instruction is of an expected type.

1 3. (Original) The method of claim 1, wherein the operand stack contains at
2 least one local variable, at least one argument that is passed as a parameter to the
3 subprogram, and an offset to the point of execution within the subprogram.

1 4. (Original) The method of claim 2, wherein the expected structure of the
2 operand stack includes a collective size of entries and the types of entries expected
3 on the operand stack at the point of execution within the subprogram.

1 ✓ 5. (Canceled).

1 6. (Original) The method of claim 4, wherein validating that the state of
2 the application snapshot on the second computing device is consistent with the
3 expected structure of the operand stack involves ensuring that the collective size
4 of entries and the types of entries on the operand stack agree with the collective
5 size of entries and the types of entries expected on the operand stack.

1 7. (Original) The method of claim 1, wherein resuming execution of the
2 application snapshot involves restarting the subprogram at the point of execution
3 within the second computing device.

1 8. (Currently amended) A computer-readable storage medium storing
2 instructions that when executed by a computer causes the computer to perform a
3 method for verifying type safety of an application snapshot, the application
4 snapshot including a state of an executing program that is moved from a first
5 computing device to a second computing device across a network in order to
6 continue execution on the second computing device, the method comprising:

7 receiving the application snapshot from the first computing device on the
8 second computing device, wherein the application snapshot contains dynamic
9 variables and defines the dynamic state of the executing program and wherein the
10 application snapshot includes a subprogram, an operand stack, and a point of
11 execution;

12 restoring the state of an object within the application snapshot on the
13 second computing device by changing a pointer from an address of the object on
14 the first computing device to an address of the object on the second computing
15 device;

16 examining the application snapshot to identify the subprogram and the
17 point of execution within the subprogram;

18 examining the subprogram to determine an expected structure of the
19 operand stack at the point of execution;

20 validating that the state of the application snapshot on the second
21 computing device is consistent with the expected structure of the operand stack;
22 and

23 if the state of the application snapshot is validated as consistent with the
24 expected structure of the operand stack, resuming execution of the application
25 snapshot on the second computing device.

1 9. (Original) The computer-readable storage medium of claim 8, wherein
2 examining the subprogram to determine the expected structure of the operand
3 stack at the point of execution involves examining the subprogram with a code
4 verifier, wherein the code verifier ensures that:

5 the subprogram does not cause the operand stack to overflow and
6 underflow;

7 a use of a local variable does not violate type safety; and

8 an argument of an instruction is of an expected type.

1 10. (Original) The computer-readable storage medium of claim 8, wherein
2 the operand stack contains at least one local variable, at least one argument that is
3 passed as a parameter to the subprogram, and an offset to the point of execution
4 within the subprogram.

1 11. (Original) The computer-readable storage medium of claim 9, wherein
2 the expected structure of the operand stack includes a collective size of entries and
3 the types of entries expected on the operand stack at the point of execution within
4 the subprogram.

1 ✓ 12. (Canceled).

1 13. (Original) The computer-readable storage medium of claim 11,
2 wherein validating that the state of the application snapshot on the second
3 computing device is consistent with the expected structure of the operand stack

4 involves ensuring that the collective size of entries and the types of entries on the
5 operand stack agree with the collective size of entries and the types of entries
6 expected on the operand stack.

1 14. (Original) The computer-readable storage medium of claim 8, wherein
2 resuming execution of the application snapshot involves restarting the subprogram
3 at the point of execution within the second computing device.

B1
Sub C1 }
1 15. (Currently amended) An apparatus that facilitates verifying type safety
2 of an application snapshot, the application snapshot including a state of an
3 executing program that is moved from a first computing device to a second
4 computing device across a network in order to continue execution on the second
5 computing device, comprising:

6 a receiving mechanism that is configured to receive the application
7 snapshot from the first computing device on the second computing device,
8 wherein the application snapshot contains dynamic variables and defines the
9 dynamic state of the executing program and wherein the application snapshot
10 includes a subprogram, an operand stack, and a point of execution;

11 an object restoring mechanism that is configured to restore the state of an
12 object within the application snapshot on the second computing device by
13 changing a pointer from an address of the object on the first computing device to
14 an address of the object on the second computing device;

15 an examination mechanism that is configured to examine the application
16 snapshot to identify the subprogram and the point of execution within the
17 subprogram wherein, the examination mechanism is configured to also examine
18 the subprogram to determine an expected structure of the operand stack at the
19 point of execution;

20 a validation mechanism that is configured to validate that the state of the
21 application snapshot on the second computing device is consistent with the
22 expected structure of the operand stack; and
23 an execution mechanism that is configured to resume execution of the
24 application snapshot on the second computing device if the state of the application
25 snapshot is validated as consistent with the expected structure of the operand
26 stack.

Sub C1 }
1 16. (Original) The apparatus of claim 15, wherein the examination
2 mechanism includes a code verifier, wherein the code verifier is configured to
3 ensure that:
4 the subprogram does not cause the operand stack to overflow and
5 underflow;
6 a use of a local variable does not violate type safety; and
7 an argument of an instruction is of an expected type.

1 17. (Original) The apparatus of claim 15, wherein the operand stack
2 contains at least one local variable, at least one argument that is passed as a
3 parameter to the subprogram, and an offset to the point of execution within the
4 subprogram.

1 18. (Original) The apparatus of claim 16, wherein the expected structure of
2 the operand stack includes a collective size of entries and the types of entries
3 expected on the operand stack at the point of execution within the subprogram.

1 ✓ 19. (Canceled).

1 20. (Original) The apparatus of claim 18, wherein the validation
2 mechanism is configured to ensure that the collective size of entries and the types
3 of entries on the operand stack agree with the collective size of entries and the
4 types of entries expected on the operand stack.

B1

Sub

C1

1 21. (Original) The apparatus of claim 15, wherein in resuming execution
2 of the application snapshot, the execution mechanism is configured to restart the
3 subprogram at the point of execution within the second computing device.

add

C1